International Journal of Current Advanced Research
ISSN: O: 2319-6475, ISSN: P: 2319-6505, Impact Factor: 6.614
Available Online at www.journalijcar.org
Volume 7; Issue 3(K); March 2018; Page No. 11192-11194
DOI: http://dx.doi.org/10.24327/ijcar.2018.11194.1931

Research Article

KEY-AGGREGATE CRYPTOSYSTEM FOR SCALABLE DATA SHARING IN CLOUD STORAGE

Samirana Acharya B., Chelika Vishnuteja., A Sai Rithwik and Cheguri Harish

CSE Dept Gurunanak Institutions Technical Campus

A R T I C L E I N F O

Article History:

Received 24th December, 2017
Received in revised form 13th January, 2018 Accepted 8th February, 2018
Published online 28th March, 2018

Key words:

Data Sharing, Cloud-storage, key-aggregate encryption, Key-Aggregate Decryption.

A B S T R A C T

Data sharing is an important functionality in cloud storage. In this article, we show how to *securely, efficiently, and flexibly* share data with others in cloud storage. Here we introduce new public-key which is used to convert the readable data into unreadable format, such process is called Encryption. The process of converting of the unreadable data to readable format is called Decryption. The novelty is that one can aggregate any set of secret keys and make them as compact as a single key, but encompassing the power of all the keys being aggregated. Here we provide formal security analysis of our schemes in the standard model and also describe other application of our schemes. In particular, our schemes give the first public-key patient-controlled encryption for flexible hierarchy; the purpose of using the Public-Key is used to protect the data from unauthorised persons.

INTRODUCTION

Cloud storage is gaining popularity recently. In enterprise settings, we see the rise in demand for data outsourcing. It is also used as a core technology behind many online services for personal applications. Nowadays, it is easy to apply for free accounts for email, photo album, file sharing and/or remote access, with storage size more than 25GB. Together with the current wireless technology, users can access almost all of their files and emails by a mobile phone in any corner of the world.

Data sharing is an important functionality in cloud storage. For example, bloggers can let their friends view a subset of their private pictures; an enterprise may grant her employees access to a portion of sensitive data. The challenging problem is how to effectively share encrypted data. Of course users can download the encrypted data from the storage, decrypt them, then send them to others for sharing, but it loses the value of cloud storage. Users should be able to delegate the access rights of the sharing data to others so that they can access these data from the server directly. However, finding an efficient and secure way to share *partial* data in cloud storage is not trivial. Below we will take Dropbox[1] as an example for illustration.

Assume that Alice puts all her private photos on Dropbox, and she does not want to expose her photos to everyone. Due to various data leakage possibility Alice cannot feel relieved by just relying on the privacy protection mechanisms provided by Dropbox, so she encrypts all the photos using her own keys before uploading. One day, Alice's friend, Bob, asks her to share the photos taken over all these years which Bob appeared in. Alice can then use the share function of Dropbox, but the problem now is how to delegate the *decryption rights* for these photos to Bob. A possible option Alice can choose is to securely send Bob the secret keys involved. Naturally, there are two extreme ways for her under the traditional encryption paradigm

- Alice encrypts all files with a single encryption key and gives Bob the corresponding secret key directly.
- Alice encrypts files with distinct keys and sends Bob the corresponding secret keys.

Encryption keys also come with two flavours - sym-metric key or asymmetric (public) key. Using symmetric encryption, when Alice wants the data to be originated from a third party, she has to give the encryption her secret key; obviously, this is not always desirable. By contrast, the encryption key and decryption key are different in public-key encryption. The use of public-key encryption gives more flexibility for our applications. For example, in enterprise settings, every employee can up-load encrypted data on the cloud storage server without the knowledge of the company's master-secret key.

Therefore, the best solution for the above problem is that Alice encrypts files with distinct public-keys, but only sends Bob a single (constant-size) decryption key. Since the decryption key should be sent via a secure channel and kept secret, small key size is always de-sirable. For example, we can not expect large storage for decryption keys in the resource-constraint devices like smart phones, smart cards or wireless sensor nodes.

Corresponding author:* **Samirana Acharya B
CSE Dept Gurunanak Institutions Technical Campus

Especially, these secret keys are usually stored in the tamper-proof memory, which is relatively expensive. The present research efforts mainly focus on minimizing the communication requirements (such as bandwidth, rounds of communication) like aggregate signature [6]. However, not much has been done about the key itself

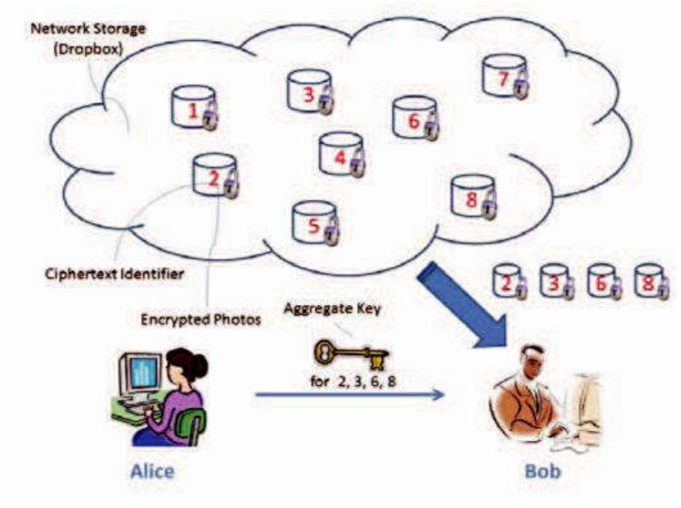

Fig 1 Alice shares files with identifiers 2, 3, 6 and 8 with Bob by sending him a single aggregate key.

We solve this problem by introducing a special type of public-key encryption which we call key-aggregate cryptosystem (KAC). In KAC, users encrypt a message not only under a public-key, but also under an identifier of ciphertext called *class*. That means the ciphertexts are further categorized into different classes. The key owner holds a master-secret called *master-secret key*, which can be used to extract secret keys for different classes. More importantly, the extracted key have can be an aggregate key which is as compact as a secret key for a single class, but aggregates the power of many such keys, i.e., the decryption power for any subset of ciphertext classes. With our solution, Alice can simply send Bob a single *aggregate key* via a secure e-mail. Bob can download the encrypted photos from Alice's Dropbox space and then use this aggregate key to decrypt these encrypted photos. The scenario is depicted in Figure 1.The sizes of ciphertext, public-key, master-secret key and aggregate key in our KAC schemes are all of constant size. The public system parameter has size linear in the number of ciphertext classes, but only a small part of it is needed each time and it can be fetched on demand from large (but non-confidential) cloud storage.Previous results may achieve a similar property featuring a constant-size decryption key, but the classes need to conform to some pre-defined hierarchical relationship. Our work is flexible in the sense that this constraint is eliminated, that is, no special relation is required between the classes We propose several concrete KAC schemes with dif-ferent security levels and extensions in this article. All constructions can be proven secure in the standard model. To the best of our knowledge, our aggregation mechanism[2] in KAC has not been investigated.

Existing System

Cryptographic key assignment schemes aim to minimize the expense in storing and managing secret keys for general cryptographic use. It proposed a method to generate a tree hierarchy of symmetric keys by using repeated evaluations of block-cipher on a fixed secret

Drawbacks in Existing System

- Increases the costs of storing and transmitting cipher text.
- The costs and complexities involved generally increase with increase with the number of the decryption keys to be shared.

Proposed System

In this method we are introducing a special type of public-key encryption which we call key-aggregate cryptosystem (KAC). In KAC, users encrypt a message not only under a public-key, but also under an identifier of ciphertext called class. That means the ciphertexts are further categorized into different classes. The key owner holds a master-secret called master-secret key.

Advantages in Proposed System

- The delegation of decryption can be efficiently implemented with the aggregate key, which is only of fixed size.
- Number of ciphertext classes is large.
- It is easy to key management.

Key-aggregate encryption

We first give the framework and definition for key-aggregate encryption. Then we describe how to use KAC in a scenario of its application in cloud storage.

Framework

A key-aggregate encryption scheme consists of five polynomial-time algorithms as follows.The data owner establishes the public system param-eter via Setup and generates a public/master-secret[3] key pair via KeyGen. Messages can be encrypted via Encrypt by anyone who also decides what ciphertext class is associated with the plaintext message to be encrypted. The data owner can use the master-secret to generate an aggregate decryption key for a set of ciphertext classes via Extract. The generated keys can be passed to delegatees securely (via secure e-mails or secure devices) Finally, any user with an aggregate key can decrypt any ciphertext provided that the ciphertext's class is contained in the aggregate key via Decrypt.

- Setup (1^λ, n): executed by the data owner to setup an account on an *untrusted* server. On input a security level parameter 1^λ and the number of ciphertext classes n (i.e., class index should be an integer bounded by 1 and n), it outputs the public system parameter param, which is omitted from the input of the other algorithms for brevity.
- KeyGen: executed by the data owner to randomly generate a public/master-secret key pair (pk, msk).
- Encrypt (pk, i, m): executed by anyone who wants to encrypt data. On input a public-key pk, an index i denoting the ciphertext class, and a message m, it outputs a ciphertext C.
- Extract (msk, S): executed by the data owner for del-egating the decrypting power for a certain set of ci-phertext classes to a delegatee. On input the master-secret key msk and a set S of indices corresponding to different classes, it outputs the aggregate key for set S denoted by K_S.

- Decrypt (K_S, S, i, C): executed by a delegatee who received an aggregate key K_S generated by Extract. On input K_S , the set S, an index i denoting the

Algorithm for Encryption and Decryption

STEP1: SETUP: compute the keys to be encrypt
STEP2: KEYGEN: *provide the function*
STEP3: ENCRYPT for a message $m \in GT$ and an index
STEP4: Extract (msk = γ, S): for the set of s of index j the aggregate key
STEP5: DECRYT: (KS, S, , C = _$c1$, $c2$, $c3$_) ot**herwise return a message**

Performance analysis

Compression Factors

For a concrete comparison, we investigate the space requirements of the tree-based key assignment approach we described in Section 3.1. This is used in the Complete Subtree scheme, which is a representative solution to the broadcast encryption problem following the well-known Subset-Cover framework.

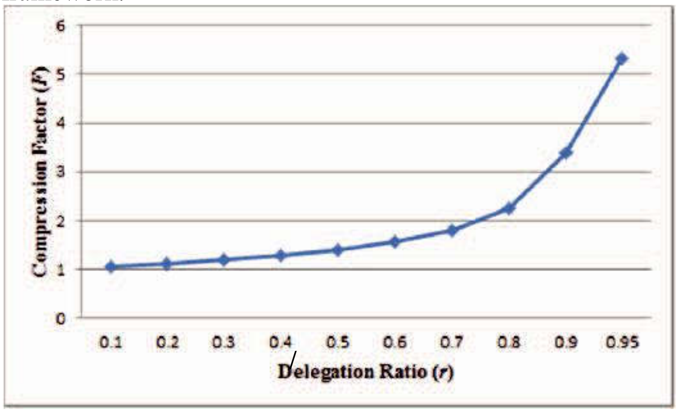

Fig 2 Compression achieved by the tree-based approach for delegating different ratio of the classes.

CONCLUSION

Data privacy is a central question of cloud storage. With more mathematical tools, cryptographic schemes are getting more versatile and often involve multiple keys for a single application. In this article, we consider how to "compress" secret keys in public-key cryptosystems which support delegation of secret keys for different ciphertext classes in cloud storage. No matter which one among the power set of classes, the delegatee can always get an aggregate key of constant size. Our approach is more flexible than hierarchical key assignment which can only save spaces if all key-holders share a similar set of privileges.

Future Work

A limitation in our work is the predefined bound of the number of maximum ciphertext classes. In cloud storage, the number of cipher texts usually grows rapidly. So we have to reserve enough ciphertext classes for the future extension

References

1. S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu, "SPICE - Simple Privacy-Preserving Identity-Management for Cloud Envi-ronment," in *Applied Cryptography and Network Security - ACNS 2012*, ser. LNCS, vol. 7341. Springer, 2012, pp. 526-543.
2. L. Hardesty, "Secure computers aren't so secure," MIT press, 2009, http://www.physorg.com/news176107396.html.
3. C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Trans. Computers*, vol. 62, no. 2, pp. 362-375, 2013.
4. B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing Shared Data on the Cloud via Security-Mediator," in *International Conference on Distributed Computing Systems - ICDCS 2013*. IEEE, 2013.
5. S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic Secure Cloud Storage with Provenance," in *Cryptog-raphy and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, ser. LNCS, vol. 6805. Springer, 2012, pp. 442-464.
6. D. Boneh0, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in *Proceedings of Advances in Cryptology - EUROCRYPT '03*, ser. LNCS, vol. 2656. Springer, 2003, pp. 416-432.
7. M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 12, no. 3, 2009.
8. J. Benaloh, M. Chase, E. Horvitz, and K. Lauter, "Patient Controlled Encryption: Ensuring Privacy of Electronic Medical Records," in *Proceedings of ACM Workshop on Cloud Computing Security (CCSW '09)*. ACM, 2009, pp. 103-114.
9. F. Guo, Y. Mu, Z. Chen, and L. Xu, "Multi-Identity Single-Key Decryption without Random Oracles," in *Proceedings of Informa-tion Security and Cryptology (Inscrypt '07)*, ser. LNCS, vol. 4990. Springer, 2007, pp. 384-398.
10. V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Com-munications Security (CCS '06)*. ACM, 2006, pp. 89-98.

How to cite this article:

Samirana Acharya B *et al* (2018) 'Key-Aggregate Cryptosystem for Scalable Data Sharing in Cloud Storage', *International Journal of Current Advanced Research,* 07(3), pp. 11192-11194. DOI: http://dx.doi.org/10.24327/ijcar.2018.11194.1931

****